CMD | FED | Y2Q2W7 | Lesbrief | PHP en OOP

Lesdoelen

- ✓ Je begrijpt waar Object geOrienteerd Programeren over gaat.
- ✓ Je begrijpt de relatie tussen een Class en een object.
- ✓ Je begrijpt het begrip Encapsulation.
- ✓ Je kunt een eenvoudig UML classdiagram lezen.
- ✓ Je kent het verschil tussen public en private members.
- ✓ Je begrijpt waartoe setters en getters dienen.
- ✓ Je begrijpt het gebruik van het keyword 'this'.
- ✓ Je begrijpt waarvoor een constructor wordt gebruikt.
- ✓ Je kunt een eenvoudig class bouwen (met properties en methods)
- ✓ Je kunt een object initialiseren en gebruiken in je script.

Inhoudsopgave

ObjectgeOriënteerd Programmeren binnen php

Inleiding OOP binnen php

De komende 3 lessen gaan we een eerste stap zetten tot het gebruik van ObjectgeOriënteerd Programmeren (OOP) binnen PHP. We behandelen de grondsbeginselen van OOP en hoe deze kunnen worden toegepast binnen PHP.

Wat bedoelen we met OOP?

OOP kan je zien als een groep samenwerkende *objecten*. Deze objecten zijn kleine zelfstandige programaatjes die allemaal hun eigen eigenschappen en mogelijkheden hebben.

Om deze objecten aan te kunnen maken moeten we *classes* definiëren. Een *class* is een stuk script met de blauwdruk van de eigenschappen (*properties*) en mogelijkheiden (*methodes*) van een object. Deze blauwdruk gebruik je vervolgens om een of meerdere *instantie(s)* aan te maken van die *class*, dit zijn individuele objecten.

Een van de belangrijkste voordelen van OOP is de inkapseling van properties en methodes. Deze inkapseling zorgt ervoor dat je bepaalde eigenschappen en mogelijkheden van een objecten kan afgeschermen of juist beschikbaar maken voor de buitenwereld, dit heet *encapsulation.*

Je kan als metafoor bijvoorbeeld auto's gebruiken. Stel dat we de class *Auto* hebben, dit is de basis bouwtekening van alle auto's. In de basis hebben alle auto's namelijk dezelfde eigenschappen (motor is aan/uit, richting, snelheid, kleur, merk, etc) en mogelijkheden (gas geven, remmen, sturen, etc.). Elke *instantie* die je aanmaakt van de class *Auto* bevat de basis eigenschappen uit de *class*. Door eigenschappen van het geinstantieerde object te veranderen, kan je bijvoorbeeld een uniek auto object aanmaken met een eigen kleur of merk. Tevens heeft elke auto zijn eigen snelheid, richting en positie.

TODO: DIAGRAM AUTO CLASS

Waarom OOP gebruiken?

Bij het tot nu toe gebruikte procedureel programmeren ga je ervan uit dat je één enkel script hebt. Hierdoor kan je vrij snel in de problemen komen binnen een complex project, vooral als het veel verschillende onderdelen bevat die samen moeten werken. Je script wordt erg lang en je moet goed opletten dat je variabelen en functies elkaar niet in de weg zitten. Als er iets veranderd dient te worden blijkt dat vaak erg lastig te zijn. Dit komt omdat het vaak onduidelijk is waar precies de afhankelijkheden liggen tussen de verschillende onderdelen van je script.

Door in deze complexere projecten OOP toe te passen kan je de functionaliteit van elk onderdeel in een eigen *class* script definiëren. Hierdoor wordt je lange script opgesplitst in kortere zelfstandige *class* scripts. In deze *classes* staan alleen de relevante methodes en properties die met het betreffende onderdeel te maken hebben. Je kan nu *instanties* (objecten) aanmaken van deze *classes* die allemaal zelfstandig opereren en hun eigen status bijhouden.

Fouten binnen je script treden vaak op binnen een enkele class of object en zijn hierdoor gemakkelijker en sneller te traceren.

Een class script schrijven in PHP

We gaan nu een relatief eenvoudige class schrijven met PHP, we nemen als voorbeeld een student. We bekijken welke eigenschappen (properties) en mogelijkheden (methodes) een student heeft.

Als je een class wilt schrijven in PHP moet je rekening houden met het volgende:

- ✓ Voor elke class maak je een nieuwe file aan.
- ✓ De naam van de file geef je als volgt '*naam*.class.php', dit is altijd lowercase.
- ✓ Je schrijft de definitie van een class altijd in *UpperCamelCase*, bijvoorbeeld *AutoMobiel.*
- ✓ De definitie van een van methode (mogelijkheid) binnen een class moet altijd een werkwoord bevatten en schrijf je in lowerCamelCase, bijvoorbeeld *toeteren*() of *openWindow*();
- ✓ De definitie van een property (eigenschap) is altijd omchrijvend en in lowercase en spaties worden vervangen door een *underscore*, bijvoorbeeld: *huidige_snelheid* of *motor_loopt.*

Een voorbeeld class script:

```php
<?php

class Student
{
	// properties
	private $nummer;
	private $first_name;
	private $last_name;

	// constructor, initialiseert het object
	public function __construct($_nummer)
	{
		$this->nummer = $_nummer;

		// we roepen de interne functie loadDetails aan
		$this->loadDetails();
	}

	// private methode om de data van de student in te laden
	private function loadDetails()
	{
		// de details van de student kunnen normaal uit een database worden geladen
		// aan de hand van het student nummer wat meegegeven is aan de constructor

		// nu zetten we ze nog even met de hand
		$this->first_name = "Martijn";
		$this->last_name = "Swart";
	}

	// publieke methode om de volledige naam van een student te krijgen
	public function getFullName()
	{
		return $this->first_name . " " . $this->last_name;
	}
}

?>
```

In het bovenstaande voorbeeld zie je een aantal dingen:

Constructor

Elke class bevat een zogenaamde *constructor*. De constructor is de eerste methode die wordt aangeroepen als er een nieuwe instantie van een class wordt aangemaakt. Hierdoor is de constructor uitermate geschikt om de begin waarden van properties te zetten.

In het voorbeeld hierboven geven we aan de constructor het student nummer mee. De constructor roept dan een methode aan om aan de hand van dit student nummer de student gegevens uit een database ophalen.

$this->

Elke methode of property van een class wordt binnen die class aangeroepen met "$this->". Dit is om aan te geven dat het gaat over een interne methode of property. Alle methodes en properties zijn beschikbaar binnen de hele class, ongeacht of ze public of private zijn.

Private of public

Binnen de class moet je aangeven of een methode of property privé (*private)* of publiek (*public)* is. Bij het ontwerpen van je class moet je goed kijken welke methodes of properties voor de buitenwereld beschikbaar moeten zijn. In bovenstaand voorbeeld zorg je ervoor dat je bijvoorbeeld niet van buitenaf de naam of nummer van de student kan wijzigen. Wel kunnen we met de public methode *getFullName* bijvoorbeeld de naam van een student opvragen.

Een object initialiseren binnen PHP

Nu we de blauwdruk hebben geschreven kunnen we deze gaan gebruiken om objecten te initialiseren. Dit gebeurd dat als volgt:

```
$user = new Student(2334224);
```

In de variabele *$user* stop ik een nieuwe instantie van de *class* Student. Het student nummer wordt meegegeven aan de constructor. De constructor laadt de details van de student direct in.

We kunnen dus de naam opvragen van de student:

```
$naam = $user->getFullName();

echo $naam;
```

Dit eenvoudige voorbeeld geeft al duidelijk de kracht aan van OOP, namelijk dat elk object zijn eigen logica afhandeld en deze kan afschermen voor andere objecten.

Opdrachten

Les opdracht: Student class uitbreiden.

TODO

Thuis opdracht: Database

Maak een ontwerp voor een database voor een blog en maak deze inclusief de tabellen aan met behulp van PHPMyAdmin. Een blog bevat verschillende soorten informatie bijvoobeeld; blog-posts, gebruikers en commentaar. Voor elk type informatie komt in een eigen tabel te staan. De tabel voor blog-posts bevat minimaal de volgende data: id, samenvatting, post, datum, tijd. Vul een aantal records in zodat we wat dummy data hebben. Je vrij in de naamgeving van de database, tabellen en velden.